Assignment 7 - Android Group Project

COMPX202-20B / COMPX242-20B

Due on **Friday 16th October at 11:30 pm**.

The goal of this assignment is to demonstrate the Android skills you have developed during this course by developing a larger application, bringing together a number of Android features.

This project should be completed by **groups of 2 students** working together, and includes planning, designing, developing, and testing an application using an Agile software engineering methodology. Each group will present their documentation and demonstrate their application running **during the last week of the semester**.

Programming Tip

Using Git with two programmers is more difficult than with one. There are two different programming approaches that you can use for this assignment:

1. Peer programming – one GitLab repo, one computer, one person making commits (recommended if you are not confident with Git)
2. Separate programming – one GitLab repo, two computers, two people committing
   - Either create a group https://docs.gitlab.com/ee/user/group/index.html#create-a-new-group or change repo permissions to "internal" and tick "allow users to request access"
   - Each user creates a "branch" to work on https://docs.gitlab.com/ee/gitlab-basics/create-branch.html
   - Merge and push your branch regularly https://docs.gitlab.com/ee/gitlab-basics/add-merge-request.html
   - Try to avoid merge conflicts – don't both work on the same part of your code at the same time!

Either programming approach is fine, but if you are not confident with Git we would **strongly** recommend using peer programming (especially since we haven't taught you how to solve merge conflicts in this course).

Note I, when setting up your Android application, you should create it using an *Empty Activity*. It should be named "AssignmentSeven", should use Java, and should use a minimum API level of *API 23: Android 6.0 (Marshmallow)*.

Note II, if you are planning to work on this assignment from both the lab computers and your home laptop/PC, make sure to create the initial Android application in the labs. You have a greater chance of your application working on both the lab computers and your home laptop/PC if you have created it in the labs first.

Note III, this assignment uses touch and gestures, which are VERY difficult to achieve with an emulator. We STRONGLY recommend that you use a real Android device for this assignment. If you do not have an Android device, you can borrow one from the FG link reception.

Assignment Description

You will be building a game-based application that utilizes most of the concepts you have learned about Android development and object structures. The basic idea is to build an application in which users 'throw' balls at targets.

- Your application should include a 'Home' screen, a 'Game Play' screen, and a 'High Score' screen.
- The 'Home' screen should show the name of your game, the names of the developers (you), and should have a 'start play' button.
- The 'Game Play' screen should be full screen and should implement the ball game described below (in the 'game play description' section). It should also have a scoring system showing a point count on screen and some kind of win/lose system so that a game can finish.
- The 'High Score' screen should show the 5 best scores achieved.

Game play description

- You should use the "fling" gesture to throw a ball at a target. You can take into account direction, speed, and possibly curve or spin to determine ball movement.
- You should limit the gesture so as not to allow the user to drag the ball all the way to a target.
- There should be one or more targets. We suggest using circles as targets because detecting collisions between circles is much easier to code than detecting collisions with more complex shapes.
- There should also be two or more obstacles, again circles will be simplest. An obstacle should affect the throw in some way. Possibilities include speed up, slow down, bounce, anti-gravity, destroy, give/take points. You may think of others. A question to keep in mind is 'how will the player know what to expect and what has happened when interacting with obstacles?'
- All visible objects (ball, target, obstacles) should be built as a class hierarchy with a common parent.

Using a Software Engineering Methodology

During this course we have looked at both the Waterfall and Agile software engineering methodologies (Week 9). This assignment should be developed using the processes involved in an Agile methodology (specifically Scrum). You should:

- Start the assignment by creating a product backlog.
- Use the product backlog to determine your sprints.
- For each sprint, you should:
    - Create a sprint backlog (requirements)
    - Outline design considerations for each task in the sprint backlog (design)
    - Develop the code for each task in the sprint backlog (development)
    - Test your code for each task in the sprint backlog (testing)

Part A: Initial planning – product backlog

The first thing you need to do is decide on the main features of your application and create a Product Backlog that lists them. Please note, a real product backlog would contain a series of user studies. Since we don't cover user studies in this paper, your product backlog can simply be a list of your application's main features.

Please note, although you are creating this product backlog at the start of the assignment, you are more than welcome to add or remove items as you see fit throughout the development process – this is the beauty of Agile! Just keep note of anything added or removed for Part F.

You do not need to commit your Product Backlog to GitLab, but you will be asked to present it during the last week of the course.

Part B: Sprint Planning – sprint backlog

The software methodology you are using is based on Scrum, which breaks a project up into time-boxed sprints. For Part B you will first need to decide how long your time-boxed sprints will run for (i.e. 3 days, 5 days, 1 week, etc.). Once you have decided on the length of your sprints, you can start selecting items from the Product Backlog to create a Sprint Backlog. You should select the number of items that you think you can get done in one sprint.

- For your first sprint, all you need to do is select some items out of your product backlog. These items should be broken into smaller tasks to create a Sprint Backlog.
- Every subsequent sprint should include any items left in the sprint backlog that you didn't get done in the last sprint, plus additional items from your product backlog (again broken down into smaller tasks).
- By the end of the project you should have multiple Sprint Backlogs – one for each time-boxed sprint.

You do not need to commit your Sprint Backlogs to GitLab, but you will be asked to present them during the last week of the course.

Please note, Part B should be completed for each sprint.

Part C: Sprint Design

Your next task is to outline any design aspects related to the items you are working on in your Sprint Backlog. For example, if you are working on the 'Home' screen, you should create a drawing that indicates the layout of the home screen; if you are working on creating your ball, target, and obstacles, you should create a class diagram that indicates the relationships between your Ball, Target, and Obstacle classes.

You do not need to commit your design documentation to GitLab, but you will be asked to present it during the last week of the course.

Please note, Part C should be completed for each task in each sprint.

Part D: Sprint Development

Your next task is to write the code to develop your application. At a minimum, you should commit your changes to GitLab at least once per sprint, but for full marks I would highly recommend committing after each task in your Sprint Backlog is completed. You will also need to demonstrate your application running during the last week of the course.

Please note, Part D should be completed for each task in each sprint.

Part E: Sprint Testing

Each time a task has been completed, it should be tested. You are not required to write any unit tests, since we have not taught you unit testing in Android. However, you should be manually testing your code as you move through your development. For Part E you are required to keep track of your manual testing.

For example: if you are working on a task to create the 'start' button on the 'Home' screen, you should test that the onclick works, by simply clicking on the button. This should be written down in a testing document. I.e. "Task X, start button: clicked button to make sure it worked correctly".

Another example: if you are working on your target collision detection, you should run the application to test that your application recognises when a ball collides with the target. Again, this should be written down in a testing document. I.e. "Task X, collision detection: ran the application, when the ball collides with the target, a toast pops up to show that collision has been detected".

You do not need to commit your testing documentation to GitLab, but you will be asked to present it during the last week of the course.

Please note, Part E should be completed for each task in each sprint.

Part F: Final Evaluation

Finally, once you have completed your application, you are required to write a brief one-page document evaluating the process. What worked well? What didn't work well? Were there features that you put in the initial Product Backlog that you didn't implement? Did you add things to the Product Backlog during development that you didn't foresee needing at the start? Were there times when you didn't finish everything planned in a sprint? How long did this assignment take – was it longer or shorter than you expected? Did you find any interesting bugs when you were testing your application?

Submitting

You do not need to submit your documentation – instead you will present it, and your application running, during the last week of the course (week 12). We will organize presentation slots closer to that time.

Make sure you have pushed all of your code changes to your GitLab repository. Ensure that you can see your changes on GitLab. It is usually a good idea to clone the repository back to a new location on your local machine to check that it uploaded correctly.

Grading

Part A – Initial planning	
5%	Product Backlog
Part B – Sprint Planning	
10%	Sprint Backlog for each sprint
Part C – Sprint Design	
15%	Class diagram for Ball, Target, and Obstacles
	UI layout for Home, Game, and High Score screens
	Outline of game play
	Other reasonable design documentation
Part D – Sprint Development	
50%	Home screen
	Game screen
	High Score screen
	Ball class
	Target class
	Obstacle class
	Ball movement
	Collision detection
	Scoring system
	Fun factor
	Reasonable commits
Part E – Sprint Testing	
10%	Reasonable testing documentation
Part F – Final Evaluation	
10%	One-page evaluation